NetSim Multi-Parameter Sweep Program

Software Recommended:

NetSim v13.0 (32/64 bit), DOT NET CORE SDK 3.1, Python 3.7.4

Project Download Link:

https://github.com/NetSim-TETCOS/Multi-Parameter-Sweeper_v13.0/archive/refs/heads/main.zip

Introduction:

When users want to *sweep* one or more parameters, they change their values between simulation runs, and compare and analyse the performance metrics from each run. NetSim multi-parameter sweeper enables users to automate the sweep process.

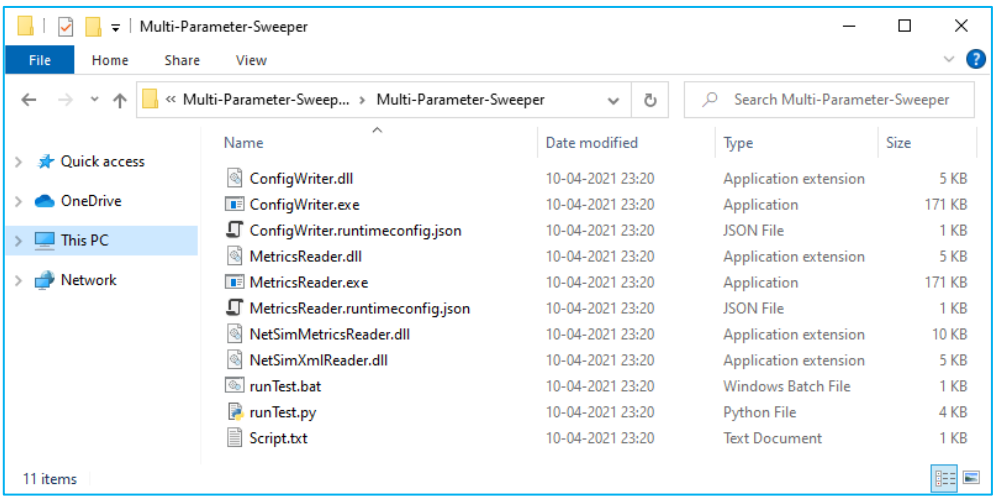
Consider an example, where a user wishes to create and simulate a network scenario for all possible values of one or more parameters in combination and analyse a set of performance metrics across the simulation runs. This is extremely time consuming to do manually using the NetSim GUI.

The multi-parameter sweep program enables users to automate the sweep process across multiple input parameters, simulate each run, save each result, and compare specific output metrics via a spreadsheet software like MS Excel.

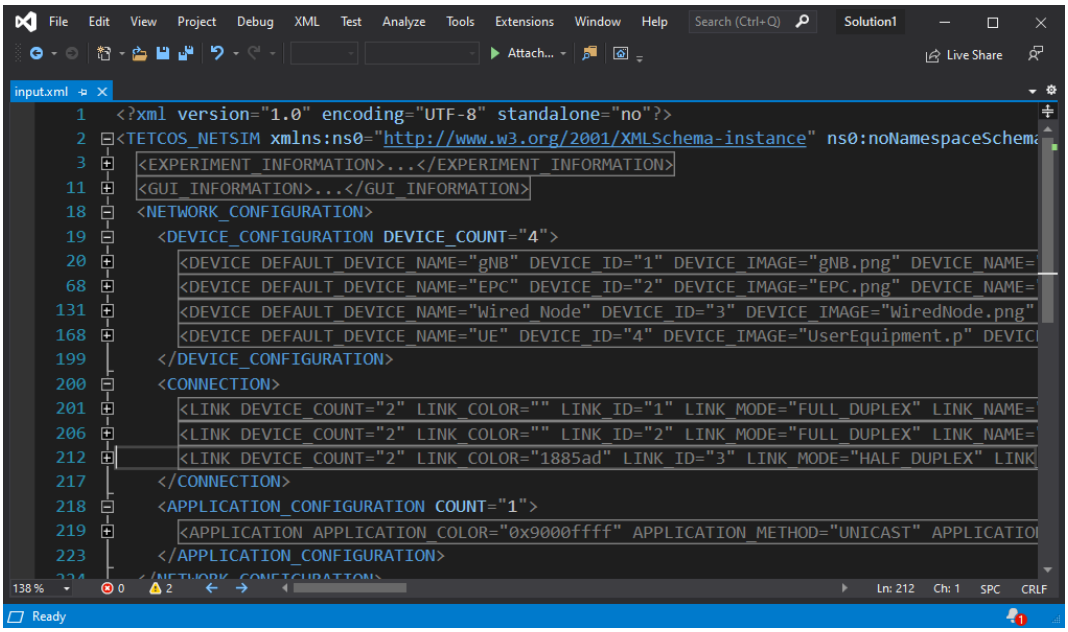
The sweep program runs NetSim via its CLI interface.

File Organization

The project directory consists of several binaries which are responsible for different tasks during a multi-parameter sweep:

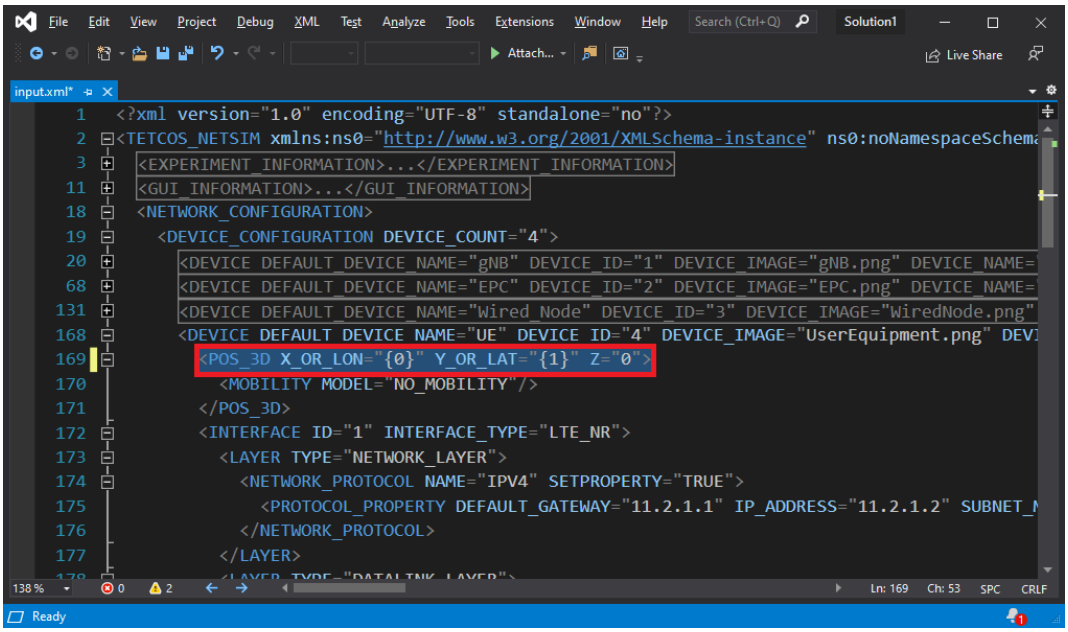


1. **input.xml:** This file contains the base NetSim network configuration that is to be simulated. This file is created by copy pasting the Configuration.netsim file that can be obtained by saving a network configuration in NetSim and renaming it to input.xml.



The values of parameters which are to be varied during each simulation run needs to be specified as {0}, {1}, {2}, etc. respectively.

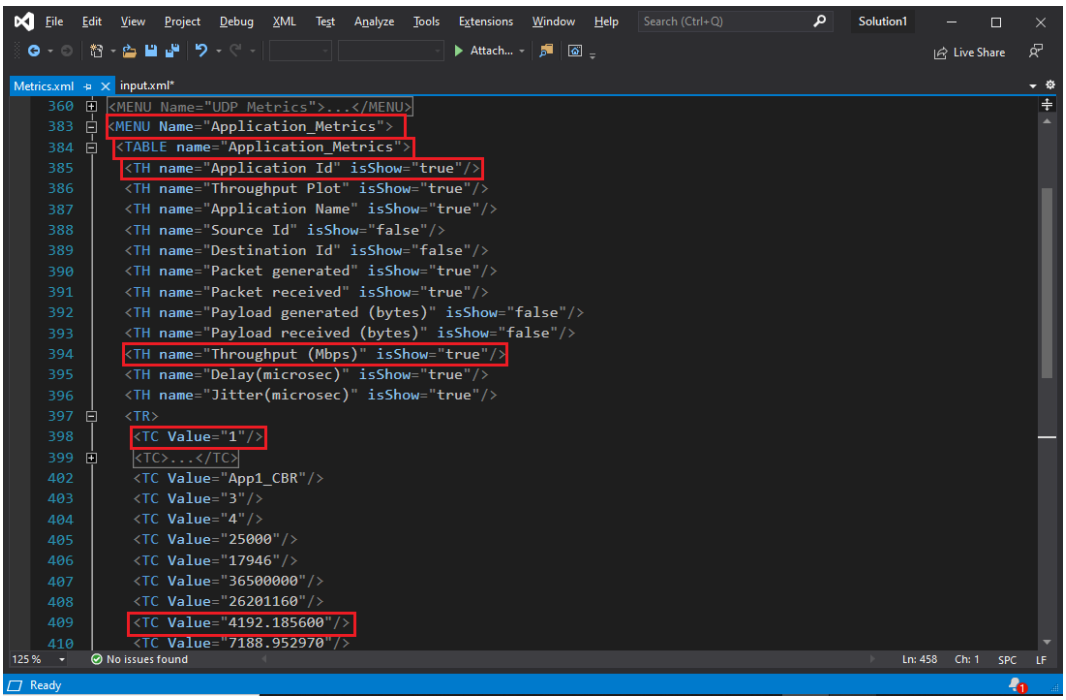
For Example, if the X and Y coordinates of a device is to be varied the values can be modified in the input.xml file as shown below:



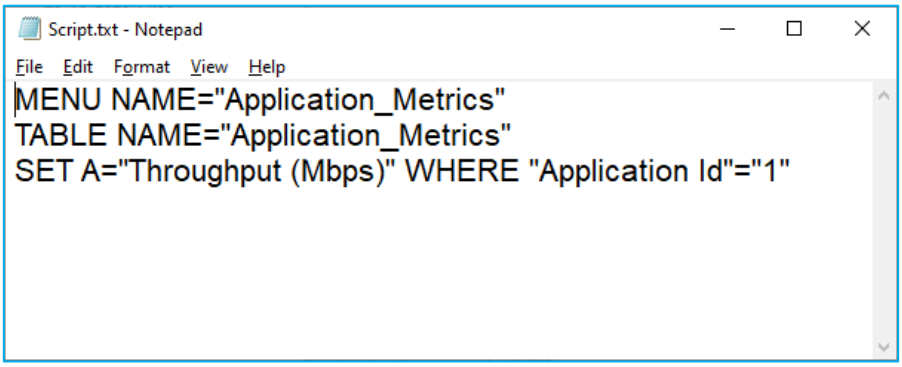
2. **Script.txt:** This file should be updated with the parameter from the output metrics of NetSim that is to be logged at the end of each simulation run for the purpose of analysis.

At the end of every simulation, NetSim generates a Metrics.xml file which contain the performance metrics written in a specific format based on which it is loaded in the results dashboard.

Each Metric is part of a results table which can be accessed using a menu in the results dashboard.
A NetSim Metrics.xml file is shown below:



For Example, if the application throughput is to be logged for each simulation run then the script file can be updated as shown below:

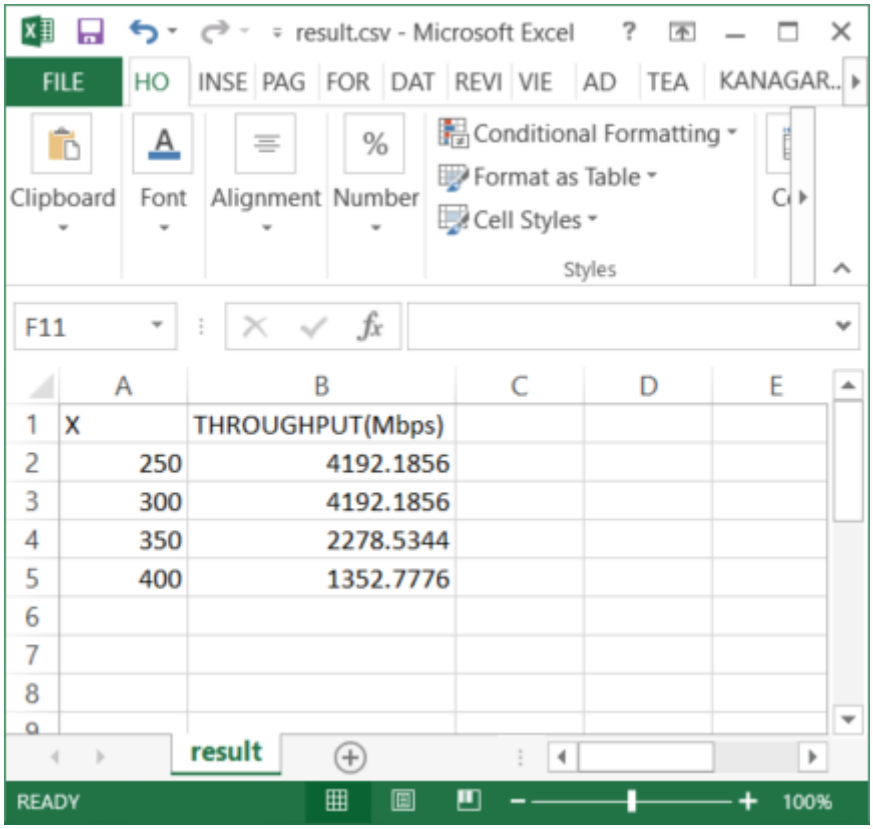


3. **ConfigWriter.exe**: This executable takes one or more command line arguments as input and generated Configuration.netsim file by replacing the arguments in place of the variable parameters specified in the input.xml file.
   If there are two variable parameters specified in the input.xml file ({0} and {1}) then two arguments need to be passed while calling ConfigWriter.exe.

4. **MetricsReader.exe**: This executable is responsible for reading the output parameter from the Metrics.xml file generated after each simulation and logging it to the results file.

   Users the Script.txt file to determine which parameter to read from the Metrics file.

If multiple parameters are to be read and logged, then the MetricsReader.exe can be called multiple times with Script.txt file having information about the parameter to be read each time.



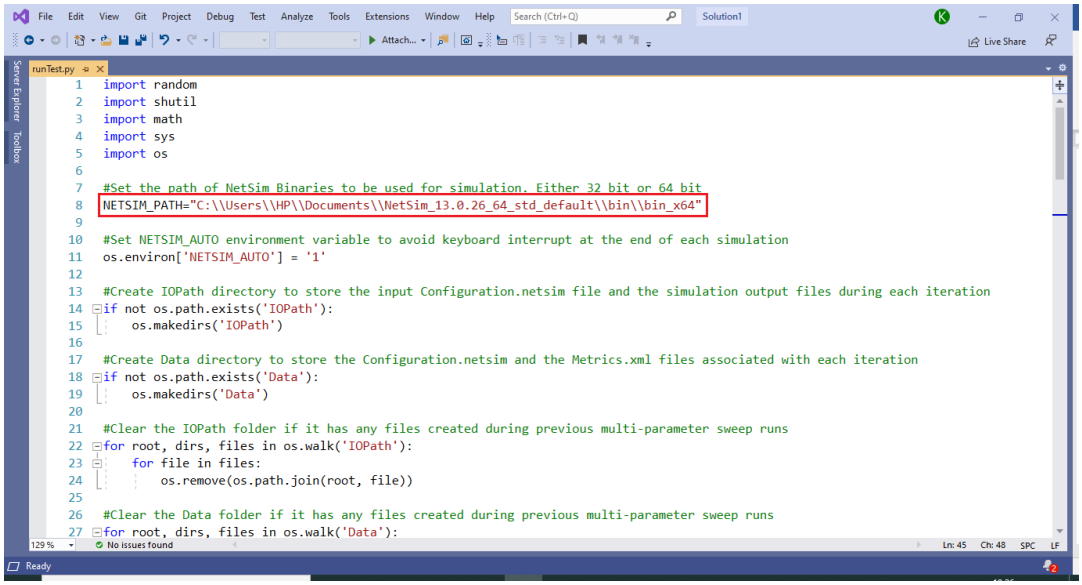
5. **Supporting DLL's**: Some the supporting files such as ConfigWriter.dll, MetricsReader.dll, NetSimMetricsReader.dll, NetSimXmlReader.dll, etc. which are present in the project folder are used by other executable such as ConfigWriter.exe and MetricsReader.exe for various purposes during a multi-parameter sweep.

6. **runTest.py** uses python programming language which is less complex and offers more flexibility as the number of input and output parameters increases.

Users can also write the script to run the multi-parameter sweep process in a preferred programming language as per the convenience.

The script can be configured to run multiple simulation iterations based on the number of parameters to be varied and the range of values of each parameter.

NETSIM_PATH variable can be set to the path of NetSim 32-bit/64-bit binaries in the install directory or workspace which is to be used to run Simulations.



For example,

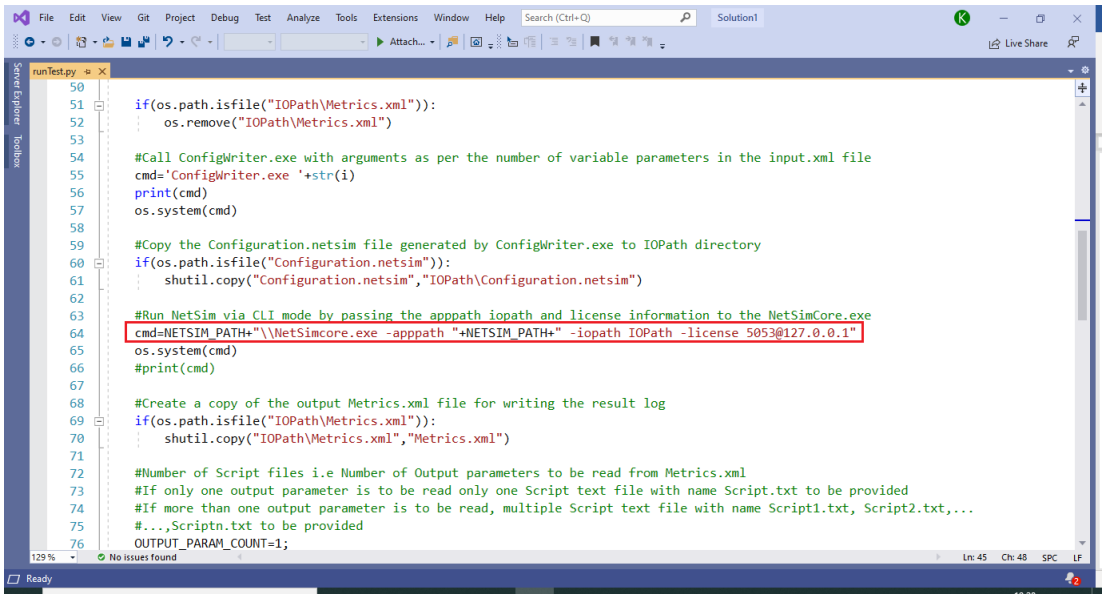
64-bit:

NETSIM_PATH="C:\\Users\\HP\\Documents\\NetSim_13.0.26_64_std_default\\bin\\bin_x64"

32-bit:

NETSIM_PATH="C:\\Users\\HP\\Documents\\NetSim_13.0.26_32_std_default\\bin\\bin_x86"

<license information> - License server details or the path of license file in case of node locked or cloud licenses



For Example,

Server based license (<port no>@<server ip address>):

cmd=NETSIM_PATH+"\\NetSimcore.exe -apppath "+NETSIM_PATH+" -iopath IOPath -license 5053@192.168.0.9”

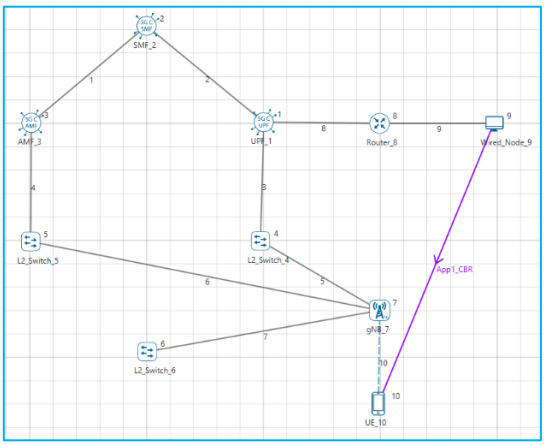
Node Locked or Cloud licenses (path of license file):

cmd=NETSIM_PATH+"\\NetSimcore.exe -apppath "+NETSIM_PATH+" -iopath IOPath -license "+"\"C:\\Program Files\\NetSim\\Standard_v13_0\\bin\""

Running a Multi-Parameter Sweep process:

Example 1: Modifying a single input parameter and logging a single output parameter

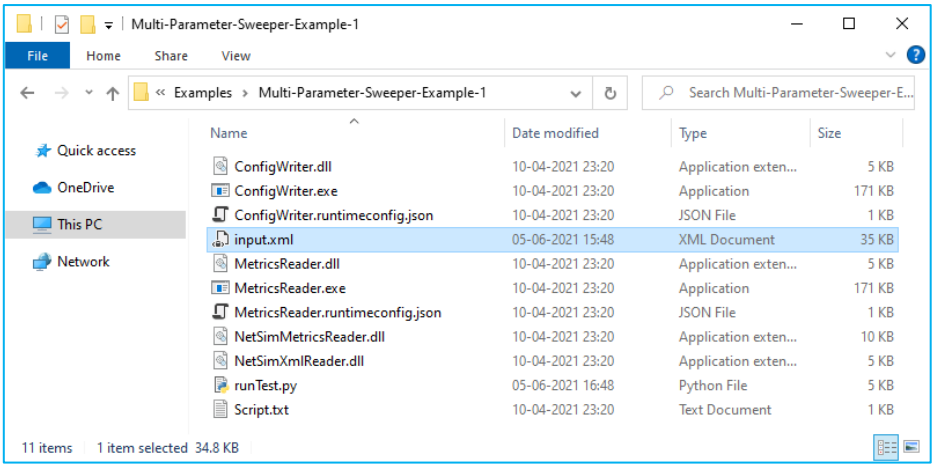
Consider the following network 5G network scenario in NetSim, comprising of a Wired Node, Router, gNB and a UE.



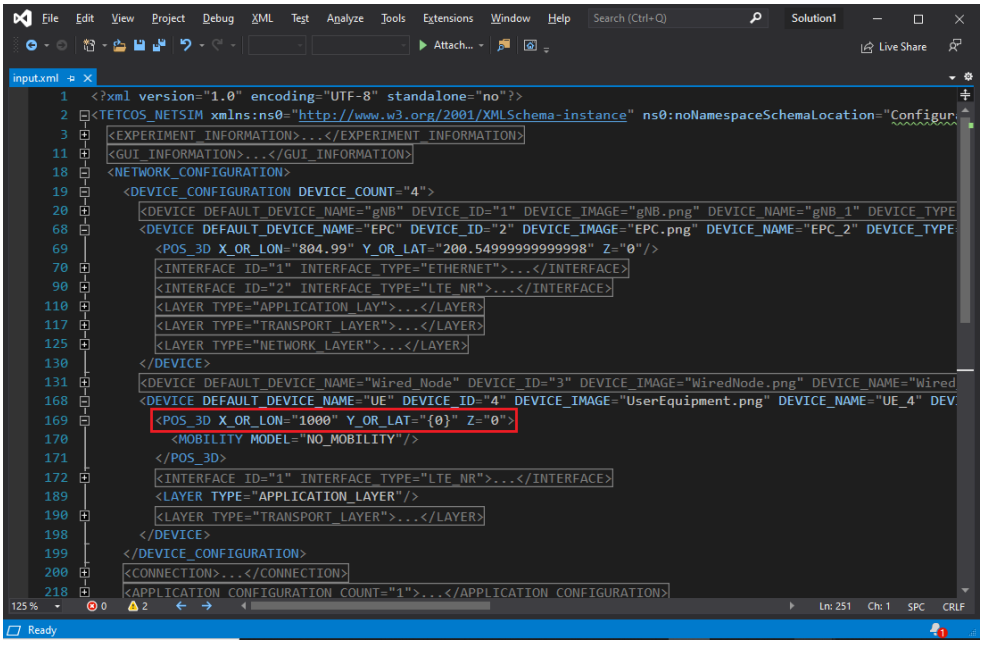
The network configuration has the initial distance between the gNB and UE as 50 meters with the gNB located at (1000,200) and UE located at (1000,250).

Multi-Parameter Sweeper is configured to run simulations for different distance between the gNB and UE by varying the UE Y coordinate value from 250 to 400 in steps of 50 meters.

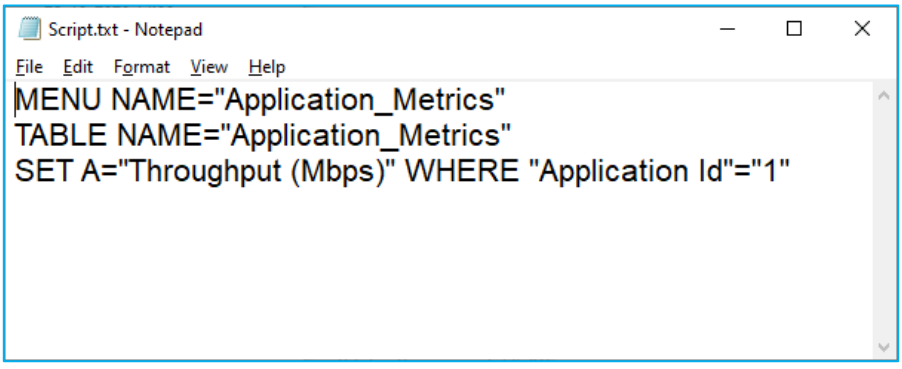
1. The network scenario is saved and the content of the Configuration.netsim file is copied to the Multi-Parameter-Sweeper directory and renamed as input.xml.
   Refer to the Example 1 directory which is part of the project folder (Multi-Parameter-Sweeper_v13.0\Examples\Multi-Parameter-Sweeper-Example-1)



The value of the Y coordinate of UE that is to be modified during each simulation run is updated ("{0}") in the configuration file as shown below:



2. The Script.txt file is updated with the details of the output parameter to be read from the Metrics.xml file and added to the result csv log file. In this case the Application throughput is to be logged for each simulation run.

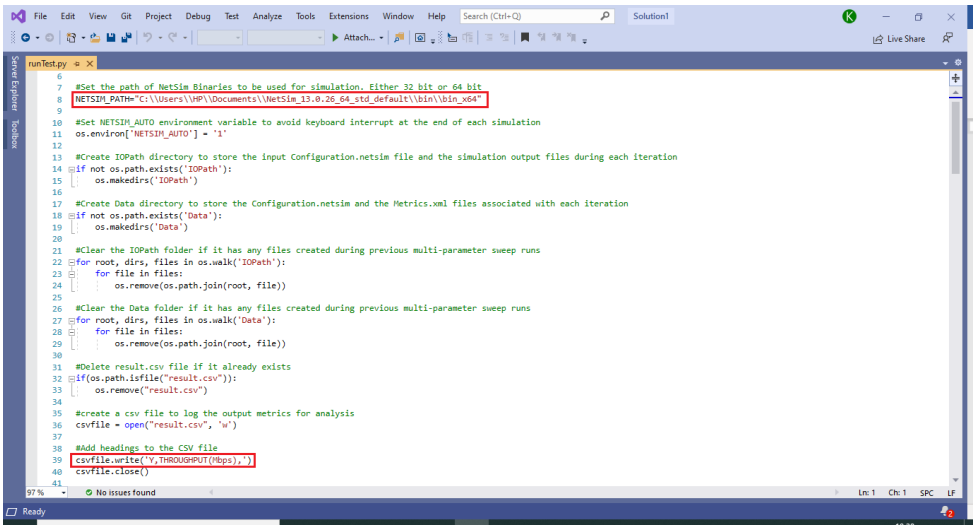


3. runTest.py is updated to pass the Y coordinate value during each iteration to generate Configuration file run simulation and update the result csv log.

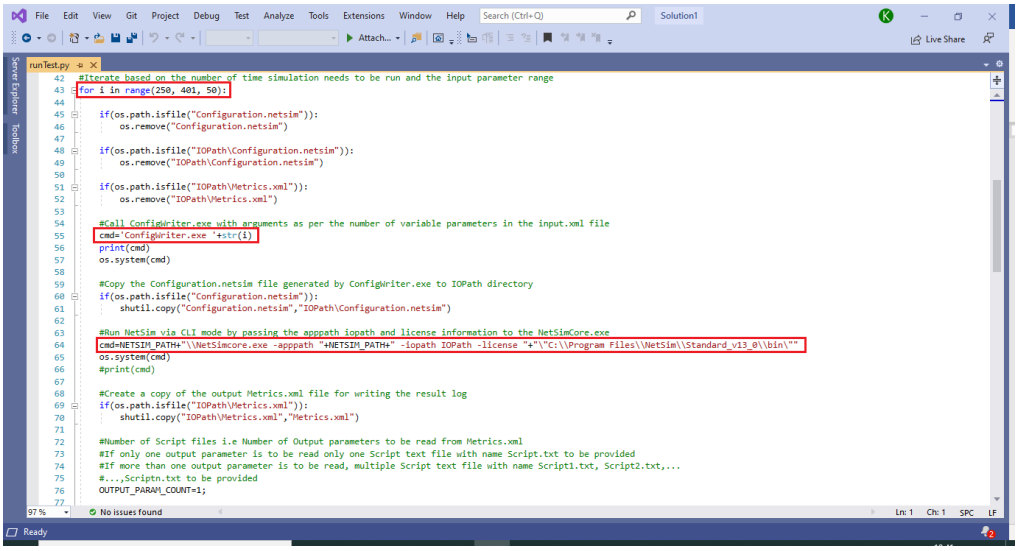
   The runTest.bat batch script modified for running simulations for different values of Y coordinates starting from 250 up to 400 in steps of 50 is shown below:

   - A result.csv file is created and added with headings Y and Throughput (Mbps).
   - For loop is set to iteratively run simulations for values starting from 250 to 400 in steps of 50.
   - The value of the parameter Y in the current iteration is written to the result log file for analysis.
   - The value of the parameter Y in the current iteration is passed as input to ConfigWriter executable to generate Configuration.netsim file for each simulation.
   - NetSim simulation is run via CLI mode by passing the apppath, iopath and license server information
   - Configuration file and Metrics file are copied and renamed appending the value of the parameter in the current iteration.

   The runTest.py python script modified for running simulations for different values of Y coordinates starting from 250 up to 400 in steps of 50 is shown below:

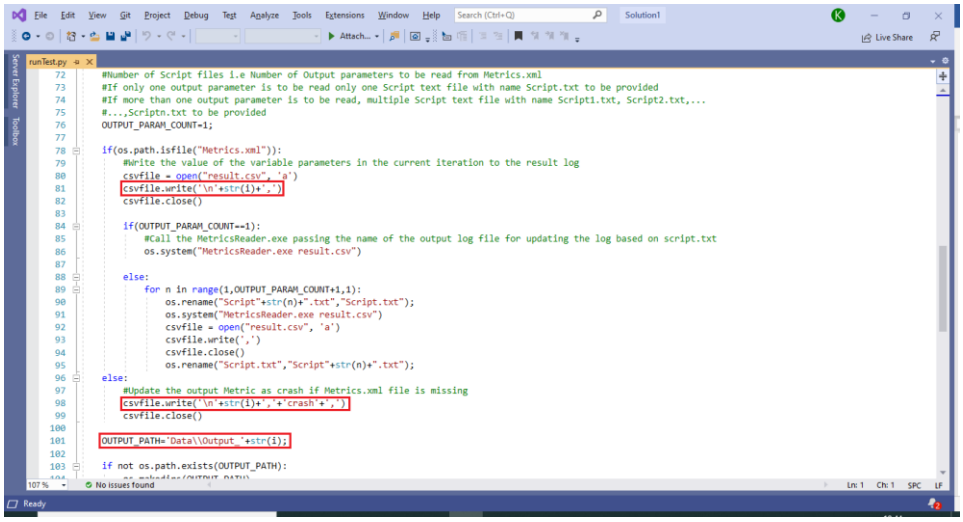


   - NETSIM_PATH variable is set to the path of NetSim 32-bit/64-bit binaries in the install directory or workspace in the system.
   - A result.csv file is created and added with headings Y and Throughput (Mbps).



```
#Iterate based on the number of time simulation needs to be run and the input parameter range
for i in range(250, 401, 50):

    if(os.path.isfile("Configuration.netsim")):
        os.remove("Configuration.netsim")

    if(os.path.isfile("IOPath\Configuration.netsim")):
        os.remove("IOPath\Configuration.netsim")

    if(os.path.isfile("IOPath\Metrics.xml")):
        os.remove("IOPath\Metrics.xml")

    #Call ConfigWriter.exe with arguments as per the number of variable parameters in the input.xml file
    cmd='ConfigWriter.exe '+str(i)
    print(cmd)
    os.system(cmd)

    #Copy the Configuration.netsim file generated by ConfigWriter.exe to IOPath directory
    if(os.path.isfile("Configuration.netsim")):
        shutil.copy("Configuration.netsim","IOPath\Configuration.netsim")

    #Run NetSim via CLI mode by passing the apppath iopath and license information to the NetSimCore.exe
    cmd=NETSIM_PATH+"\\NetSimcore.exe -apppath "+NETSIM_PATH+" -iopath IOPath -license "+"\"C:\\Program Files\\NetSim\\Standard_v13_0\\bin\\""
    os.system(cmd)
    #print(cmd)

    #Create a copy of the output Metrics.xml file for writing the result log
    if(os.path.isfile("IOPath\Metrics.xml")):
        shutil.copy("IOPath\Metrics.xml","Metrics.xml")

    #Number of Script files i.e Number of Output parameters to be read from Metrics.xml
    #If only one output parameter is to be read only one Script text file with name Script.txt to be provided
    #If more than one output parameter is to be read, multiple Script text file with name Script1.txt, Script2.txt,...
    #...,Scriptn.txt to be provided
    OUTPUT_PARAM_COUNT=1;
```

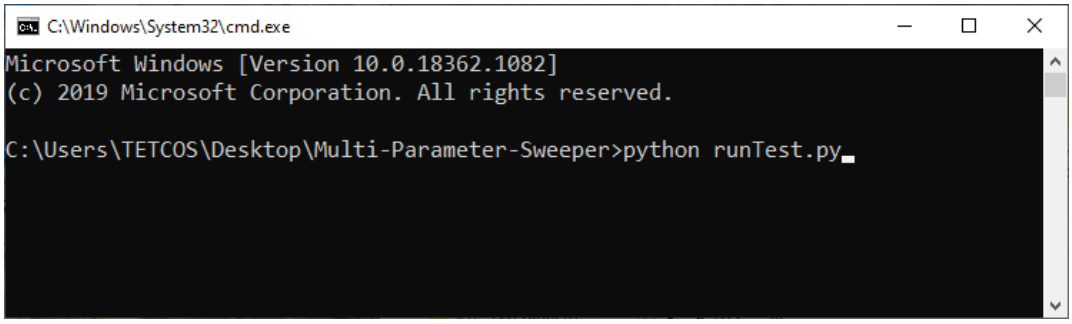
- For loop is set to iteratively run simulations for values starting from 250 to 400 in steps of 50.
- The value of the parameter Y in the current iteration is passed as input to ConfigWriter executable to generate Configuration.netsim file for each simulation.
- NetSim simulation is run via CLI mode by passing the apppath, iopath and license server information.



```
    #Number of Script files i.e Number of Output parameters to be read from Metrics.xml
    #If only one output parameter is to be read only one Script text file with name Script.txt to be provided
    #If more than one output parameter is to be read, multiple Script text file with name Script1.txt, Script2.txt,...
    #...,Scriptn.txt to be provided
    OUTPUT_PARAM_COUNT=1;

    if(os.path.isfile("Metrics.xml")):
        #Write the value of the variable parameters in the current iteration to the result log
        csvfile = open("result.csv", 'a')
        csvfile.write('\n'+str(i)+',')
        csvfile.close()

        if(OUTPUT_PARAM_COUNT==1):
            #Call the MetricsReader.exe passing the name of the output log file for updating the log based on script.txt
            os.system("MetricsReader.exe result.csv")

        else:
            for n in range(1,OUTPUT_PARAM_COUNT+1,1):
                os.rename("Script"+str(n)+".txt","Script.txt");
                os.system("MetricsReader.exe result.csv")
                csvfile = open("result.csv", 'a')
                csvfile.write(',')
                csvfile.close()
                os.rename("Script.txt","Script"+str(n)+".txt");
    else:
        #Update the output Metric as crash if Metrics.xml file is missing
        csvfile.write('\n'+str(i)+','+'crash'+',')
        csvfile.close()

    OUTPUT_PATH='Data\\Output_'+str(i);

    if not os.path.exists(OUTPUT_PATH):
```

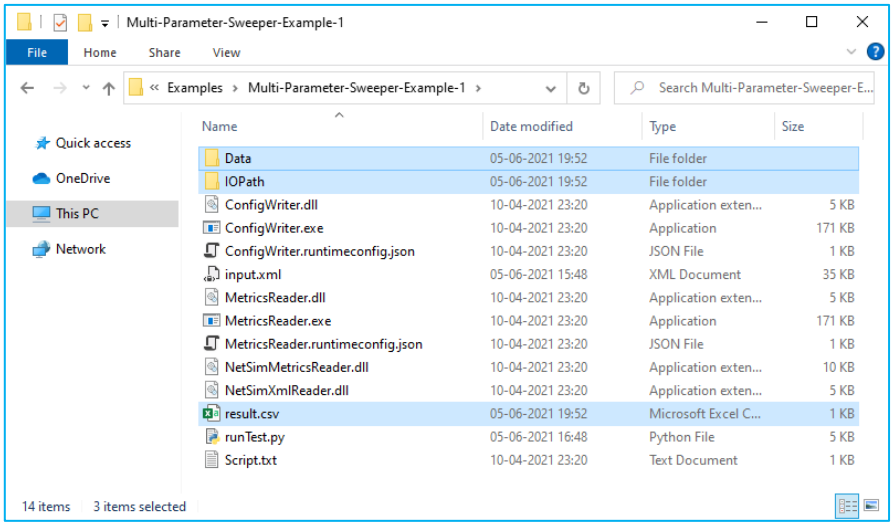
- The value of the parameter Y in the current iteration is written to the result log file for analysis.
- Configuration file and Metrics file are copied and renamed appending the value of the parameter in the current iteration.

4. Multi-Parameter Sweeping process is started by opening command prompt in the directory of the Multi-Parameter-Sweeping project and starting the python script as shown below:

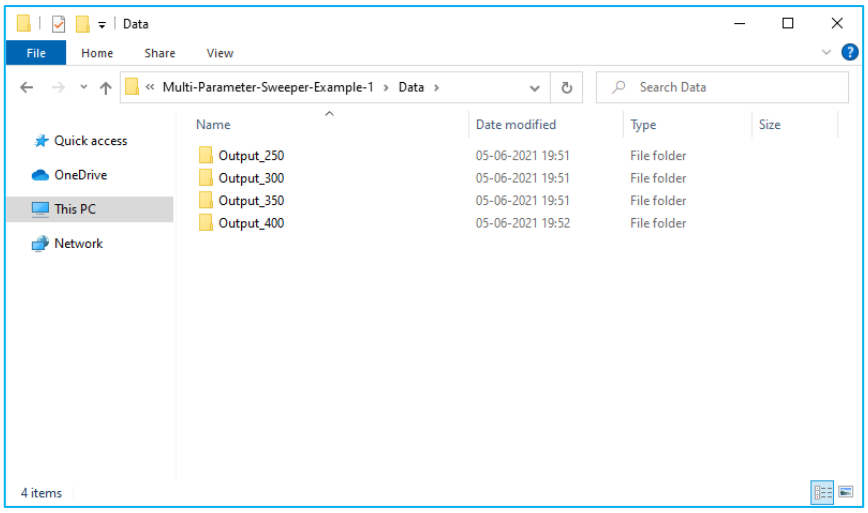


This starts the Multi-Parameter-Sweeping process which runs NetSim simulations iteratively for different values of Y parameter of UE.

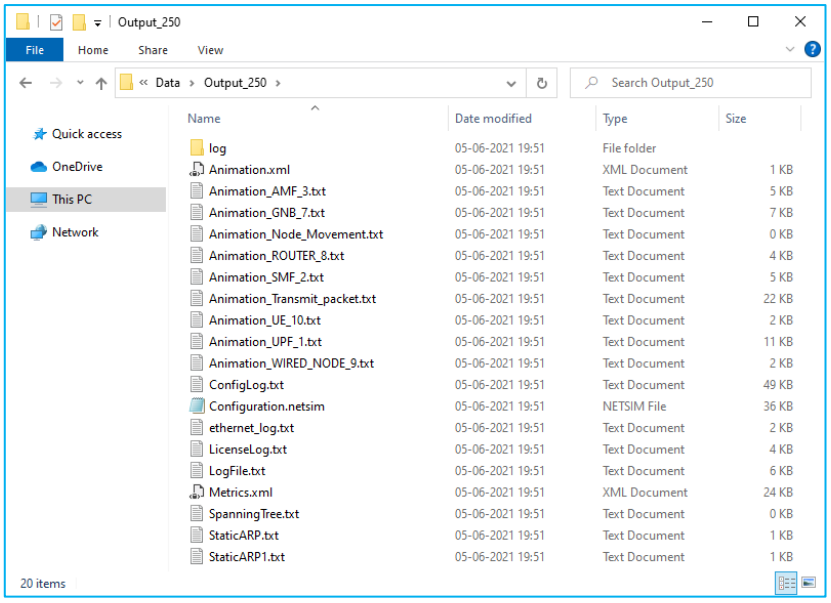
At the end of the process the Multi-Parameter-Sweeping folder will have the following file and folders created:



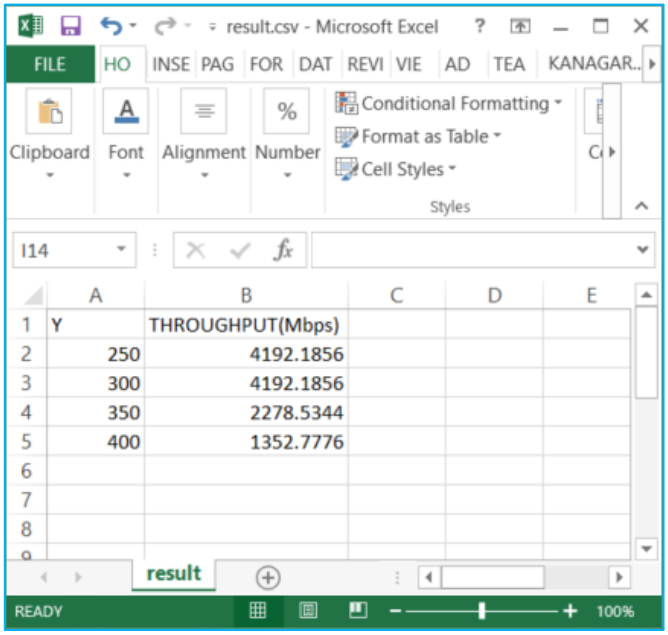
- **Data**: Contains multiple folders corresponding to each simulation run, with its name including the value of the parameters in that iteration.



- Each folder contains the all the output files associated with the simulation run.



- **IOPath**: Used for storing the Configuration.netsim file and the simulation files generated during each simulation run.
- **Result.csv**: This is the output log which contains the parameter varied during each simulation run and the output parameter associated with each run.



Varying multiple network parameters:

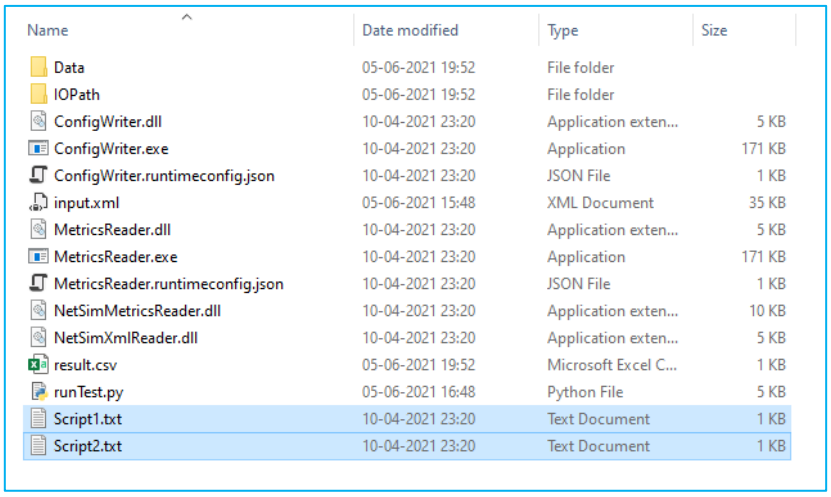
In order to vary multiple network parameters during the multi-parameter sweep process each parameter in the input.xml file can be modified as {0},{1},{2},{3},..{n} respectively.

Logging multiple output parameters:

Each output parameter that is to be logged should be part of the Script.txt file. However, the Script.txt file should contain only the details of one output parameter during the call to MetricsReader.exe.

To log multiple parameters, multiple script files can be used. If n output parameters are to be logged, then there can be script1.txt, script2.txt, script.txt in the sweeper folder.

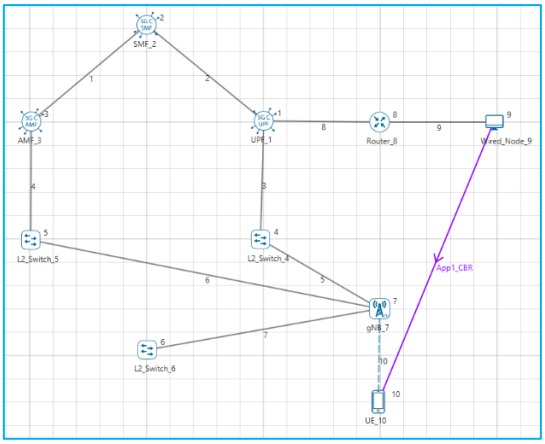
For Example, there can be two Script files as shown below:



Name	Date modified	Type	Size
Data	05-06-2021 19:52	File folder	
IOPath	05-06-2021 19:52	File folder	
ConfigWriter.dll	10-04-2021 23:20	Application exten...	5 KB
ConfigWriter.exe	10-04-2021 23:20	Application	171 KB
ConfigWriter.runtimeconfig.json	10-04-2021 23:20	JSON File	1 KB
input.xml	05-06-2021 15:48	XML Document	35 KB
MetricsReader.dll	10-04-2021 23:20	Application exten...	5 KB
MetricsReader.exe	10-04-2021 23:20	Application	171 KB
MetricsReader.runtimeconfig.json	10-04-2021 23:20	JSON File	1 KB
NetSimMetricsReader.dll	10-04-2021 23:20	Application exten...	10 KB
NetSimXmlReader.dll	10-04-2021 23:20	Application exten...	5 KB
result.csv	05-06-2021 19:52	Microsoft Excel C...	1 KB
runTest.py	05-06-2021 16:48	Python File	5 KB
Script1.txt	10-04-2021 23:20	Text Document	1 KB
Script2.txt	10-04-2021 23:20	Text Document	1 KB

Example 2: Modifying multiple input parameters and logging multiple output parameter

Consider the following network 5G network scenario in NetSim, comprising of a Wired Node, Router, gNB and a UE.



Properties configured in the LTE_NR interface of the gNB is shown in the table below:

Interface(5G_RAN) Properties	
Tx_Power(dBM)	40
Tx_Antenna_Count	8

Rx_Antenna_Count	4
CA_Type	Single Band
CA_Configuration	n78
CA_Count	1
MU	0
Channel Bandwidth (MHz)	10
PRB Count	52
MCS Table	QAM64
CQI Table	Table 1
X_Overhead	XOH0
DL UL Ratio	4:1
Outdoor Scenario	Rural Macro
LOS Mode	Standard
Wireless Link Properties	
Channel Characteristics	No_Pathloss
Wired Link Properties	
Link Speed (Mbps)	10000
BER	0
Propagation Delay (µs)	0
Application Properties	
Packet Size (Byte)	1460
Inter Arrival Time (µs)	166
Generation Rate (Mbps)	100
Transport Control	UDP
Start Time (s)	1
QoS	BE
Simulation Parameters	
Simulation Time (s)	1.1

Traffic is generated at a rate of 70 Mbps and upon running simulation, the throughput achieved is 59.95 Mbps.

We now find the max throughput for each possible bandwidth; Tx Antenna count and Rx Antenna count combination varying the generation rate based accordingly.

Two more parameters to be taken care include, the PRB Count and Guard Band (KHz) which vary with respect to the bandwidth.

Input Variables	Value Range
Channel Bandwidth (MHz)	10,15,20,25,30,40,50
Tx_Antenna_Count	1,2,4,8,16,32,64,128
Rx_Antenna_Count	1,2,4,8,16
PRB Count	52,79,106,133,160,216,270
Guard Band (KHz)	312.5,382.5,452.5,522.5,592.5,552.5,692.5
Reference Inter Arrival Time (Microseconds)	166
Reference Bandwidth	10
Reference DL MIMO Layer Count	2

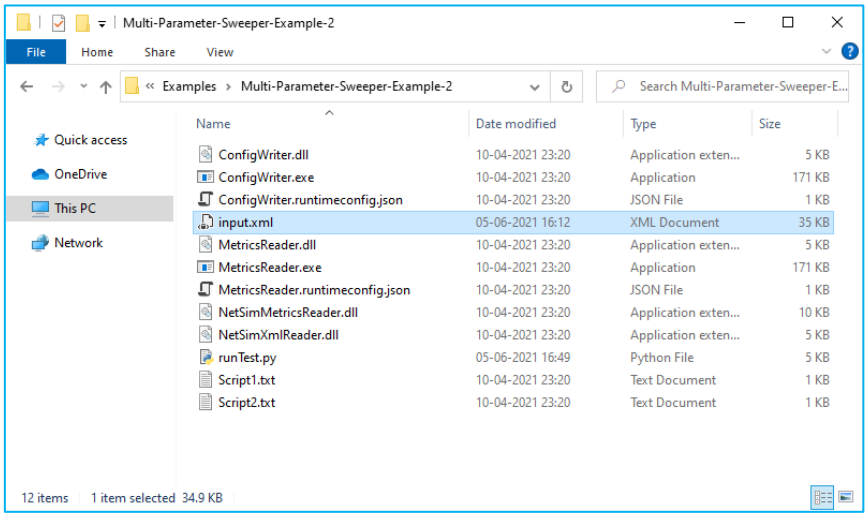
Inter Arrival Time for each case is calculated based on the Reference IAT Bandwidth and DL MIMO Layer Count as shown below:

$$Inter\ Arrival\ Time\ (Micro\ Seconds) = \frac{Ref\ IAT}{\left(\frac{Curr\ BW}{Ref\ BW}\right) * \left(\frac{Curr\ DL\ MIMO\ Count}{Ref\ DL\ MIMO\ Count}\right)}$$

For E.g. In case of Bandwidth of 20 MHz and DL MIMO Count of 4 inter arrival time is

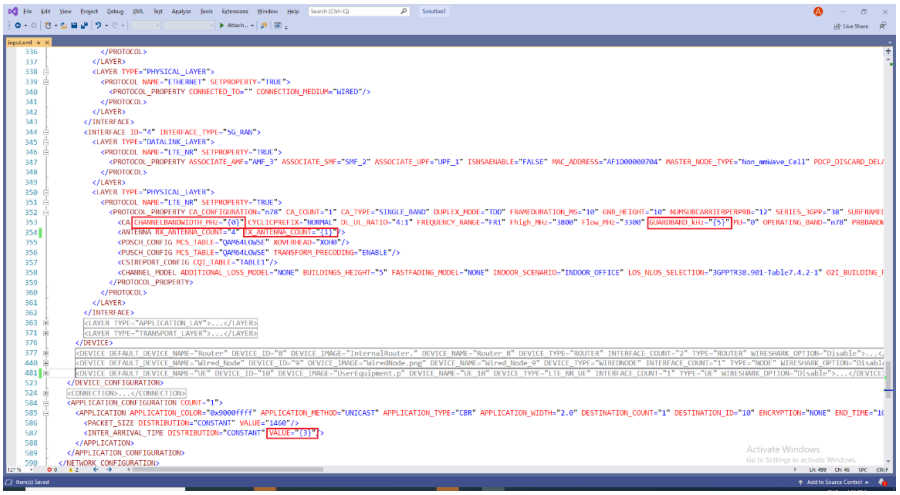
$$Inter\ Arrival\ Time\ (Micro\ Seconds) = \frac{166}{\left(\frac{20}{10}\right) * \left(\frac{4}{2}\right)} = 41.5\ Mbps$$

1. The network scenario is saved and the content of the Configuration.netsim file is copied to the Multi-Parameter-Sweeper directory and renamed as input.xml.
2. Refer to the Example 2 directory which is part of the project folder (Multi-Parameter-Sweeper_v13.0\Examples\Multi-Parameter-Sweeper-Example-2)

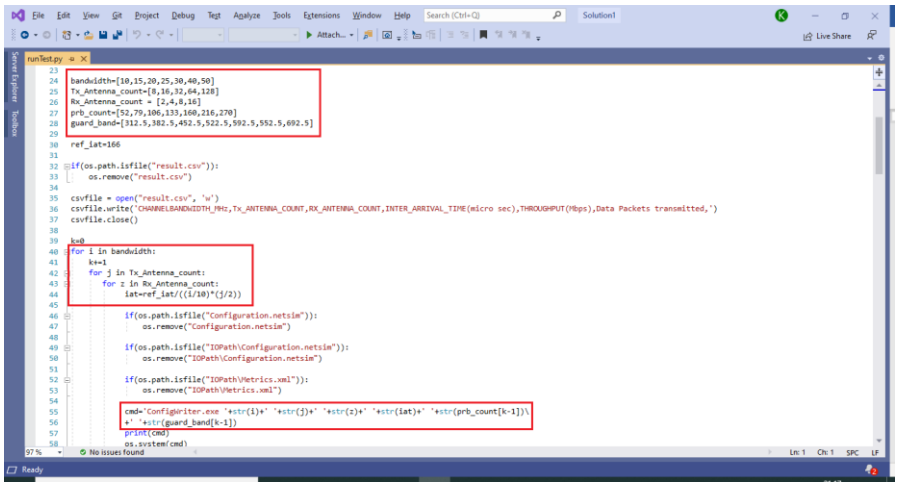


3. In the Input.xml file the value of the input variables are modified as shown in the table below:

Input Variables	
Channel Bandwidth (MHz)	{0}
Tx Antenna Count	{1}
Rx Antenna Count	{2}
Inter Arrival Time (Microseconds)	{3}
PRB Count	{4}
Guard Band (KHz)	{5}



4. The python script runTest.py is modified to run simulation for all possible combinations of Bandwidth and Tx Antenna Count and Rx Antenna Count with the respective values of Guard Band, PRB Count and the IAT that is calculated.

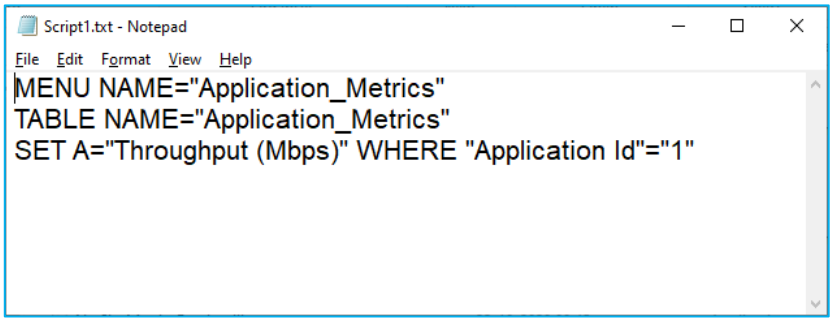


5. Multiple parameters are read from the Metrics.xml file and logged in the results.csv file along with the input parameters such as CHANNELBANDWIDTH_MHz, TX_ANTENNA_COUNT, RX_ANTENNA_COUNT, INTER_ARRIVAL_TIME (micro sec).

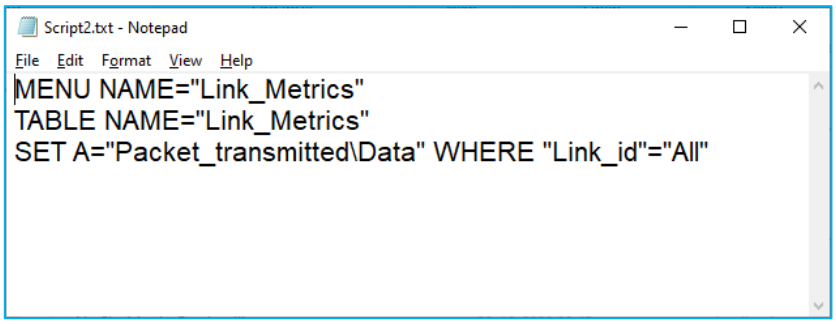
Output Parameters
Throughput (Mbps)
Data Packets transmitted

6. Two script text files namely Script1.txt and Script2.txt are created with information to read each of the parameters from the Metrics.xml file. The variable OUTPUT_PARAM_COUNT is set to 2 as per the number of Script files.

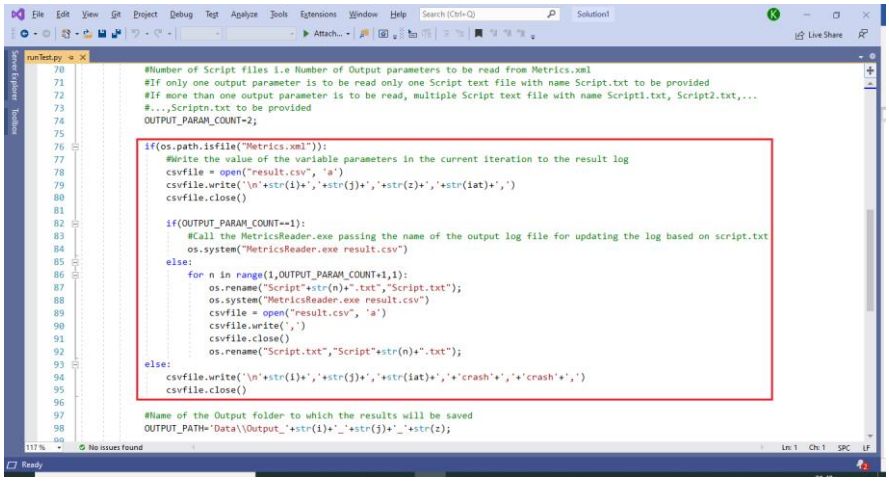
Script1.txt



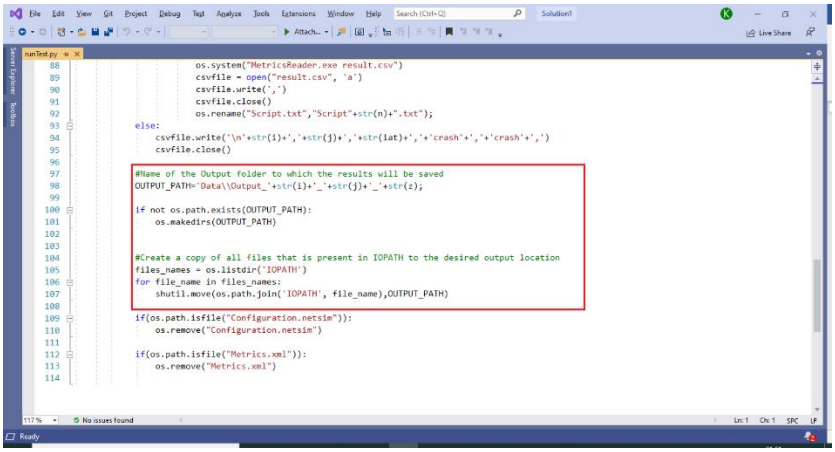
Scritp2.txt



7. In the python script runTest.py, MetricsReader is called to log each parameter specified in the script text files separating the entries with a comma (“,”). If simulation crashes, without generating the output Metrics.xml, then “crash” message is written to the log for each output parameter. The input parameters that were varied during each simulation run are also logged in the results.csv file.



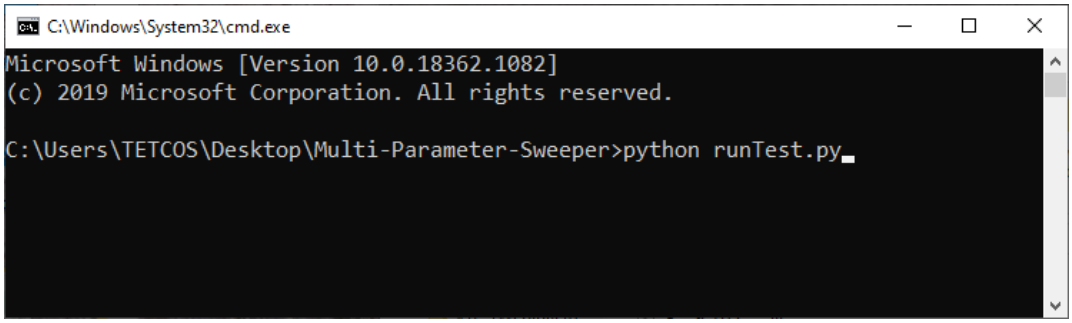
8. The simulation Configuration file and all the output files associated with each simulation run is saved to folders with name including the bandwidth and DL MIMO count values that were used during each simulation run.



```
                os.system("MetricsReader.exe result.csv")
                csvfile = open("result.csv", 'a')
                csvfile.write(',')
                csvfile.close()
                os.rename("Script.txt","Script"+str(n)+".txt");
        else:
            csvfile.write('\n'+str(i)+','+str(j)+','+str(iat)+','+'crash'+','+'crash'+',')
            csvfile.close()

            #Name of the Output folder to which the results will be saved
            OUTPUT_PATH='Data\\Output_'+str(i)+'_'+str(j)+'_'+str(z);

            if not os.path.exists(OUTPUT_PATH):
                os.makedirs(OUTPUT_PATH)


            #Create a copy of all files that is present in IOPATH to the desired output location
            files_names = os.listdir('IOPATH')
            for file_name in files_names:
                shutil.move(os.path.join('IOPATH', file_name),OUTPUT_PATH)

            if(os.path.isfile("Configuration.netsim")):
                os.remove("Configuration.netsim")

            if(os.path.isfile("Metrics.xml")):
                os.remove("Metrics.xml")
```

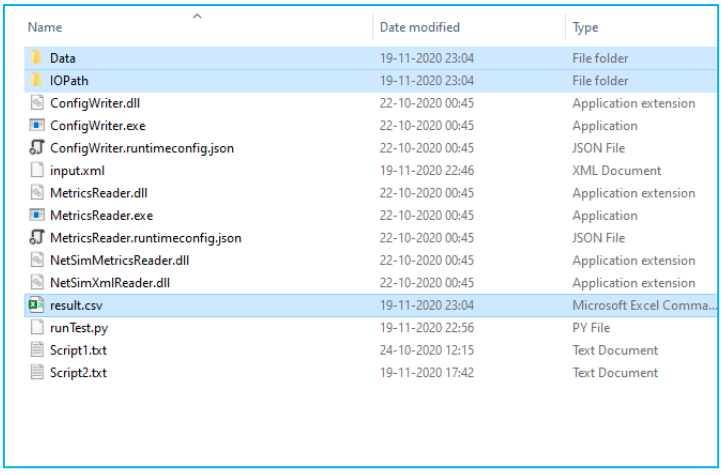
9. Multi-Parameter Sweeping process is started by opening command prompt in the directory of the Multi-Parameter-Sweeping project and starting the python script as shown below:

Python Script:



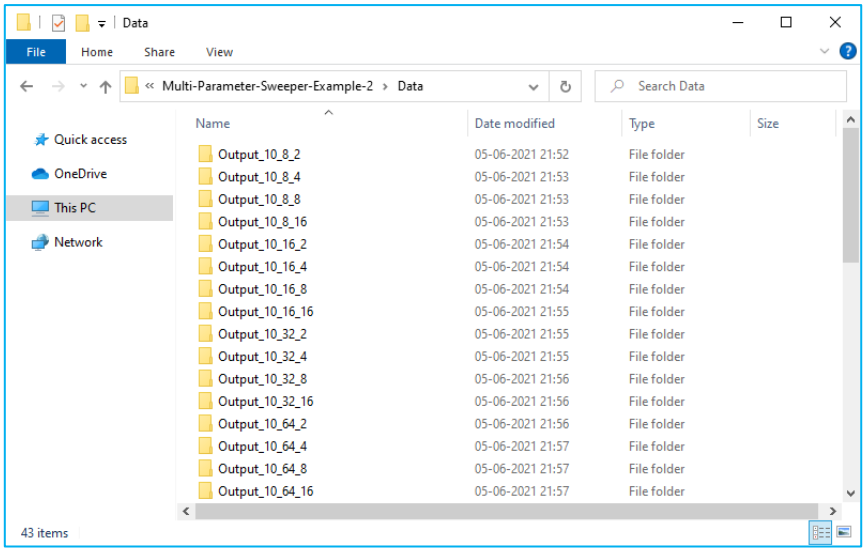
This starts the Multi-Parameter-Sweeping process which runs NetSim simulations iteratively for different combinations of input parameters.

At the end of the process the Multi-Parameter-Sweeping folder will have the following file and folders created:



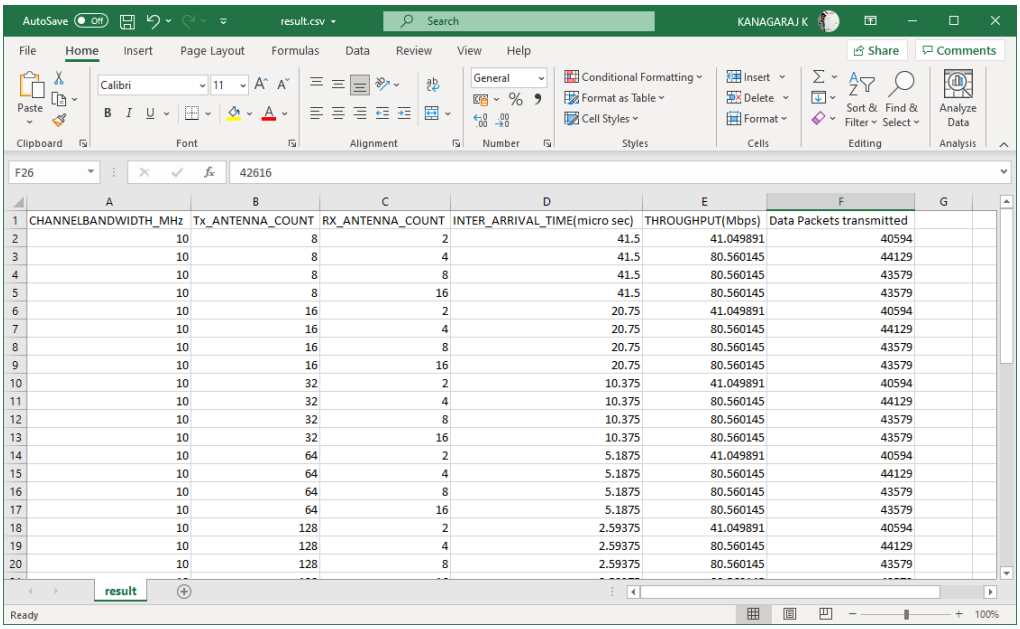
Name	Date modified	Type
Data	19-11-2020 23:04	File folder
IOPath	19-11-2020 23:04	File folder
ConfigWriter.dll	22-10-2020 00:45	Application extension
ConfigWriter.exe	22-10-2020 00:45	Application
ConfigWriter.runtimeconfig.json	22-10-2020 00:45	JSON File
input.xml	19-11-2020 22:46	XML Document
MetricsReader.dll	22-10-2020 00:45	Application extension
MetricsReader.exe	22-10-2020 00:45	Application
MetricsReader.runtimeconfig.json	22-10-2020 00:45	JSON File
NetSimMetricsReader.dll	22-10-2020 00:45	Application extension
NetSimXmlReader.dll	22-10-2020 00:45	Application extension
result.csv	19-11-2020 23:04	Microsoft Excel Comma...
runTest.py	19-11-2020 22:56	PY File
Script1.txt	24-10-2020 12:15	Text Document
Script2.txt	19-11-2020 17:42	Text Document

10. **Data**: The Data directory contains multiple output folders with the output files associated with each simulation run.



- **IOPath**: Used for storing the Configuration.netsim file and the simulation files generated during each simulation run.

- **Result.csv**: This is the output log which contains the parameter varied during each simulation run and the output parameter associated with each run.



	A	B	C	D	E	F	G
1	CHANNELBANDWIDTH_MHz	Tx_ANTENNA_COUNT	RX_ANTENNA_COUNT	INTER_ARRIVAL_TIME(micro sec)	THROUGHPUT(Mbps)	Data Packets transmitted	
2	10	8	2	41.5	41.049891	40594	
3	10	8	4	41.5	80.560145	44129	
4	10	8	8	41.5	80.560145	43579	
5	10	8	16	41.5	80.560145	43579	
6	10	16	2	20.75	41.049891	40594	
7	10	16	4	20.75	80.560145	44129	
8	10	16	8	20.75	80.560145	43579	
9	10	16	16	20.75	80.560145	43579	
10	10	32	2	10.375	41.049891	40594	
11	10	32	4	10.375	80.560145	44129	
12	10	32	8	10.375	80.560145	43579	
13	10	32	16	10.375	80.560145	43579	
14	10	64	2	5.1875	41.049891	40594	
15	10	64	4	5.1875	80.560145	44129	
16	10	64	8	5.1875	80.560145	43579	
17	10	64	16	5.1875	80.560145	43579	
18	10	128	2	2.59375	41.049891	40594	
19	10	128	4	2.59375	80.560145	44129	
20	10	128	8	2.59375	80.560145	43579	